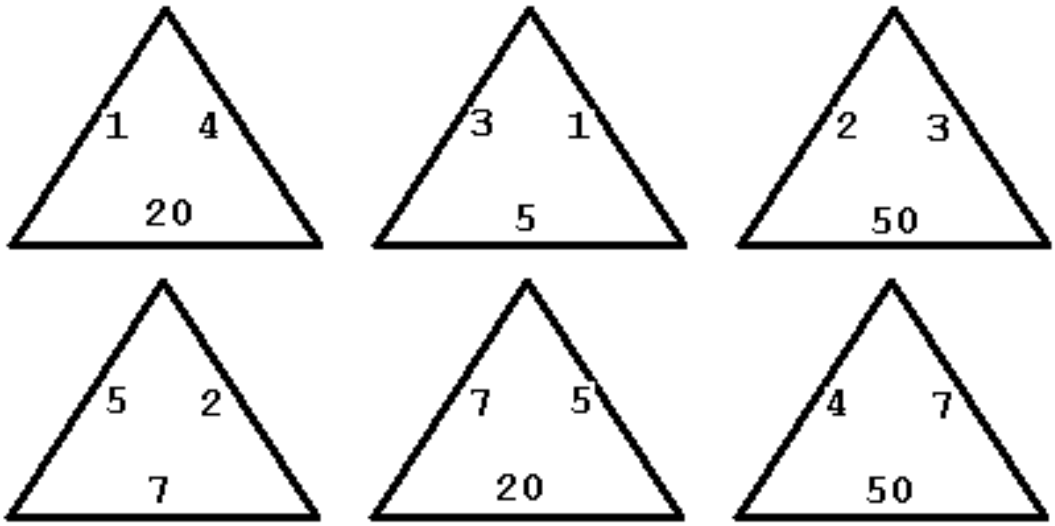
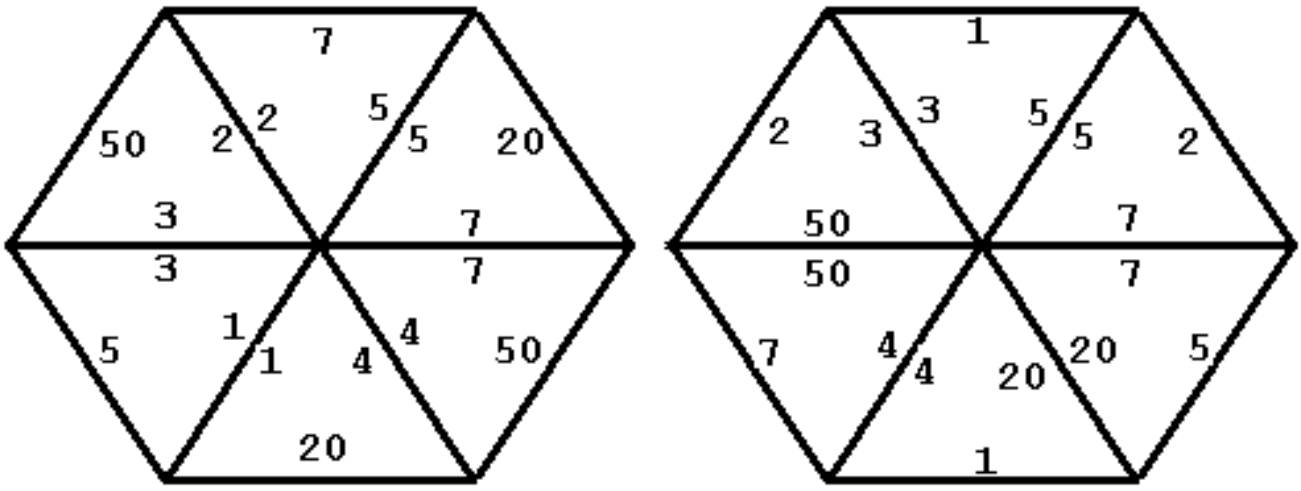
Problem A: The Triangle Game

Source file: `triangle.{c, cpp, java, pas}`
Input file: `triangle.in`
Output file: `triangle.out`



In the triangle game you start off with six triangles numbered on each edge, as in the example above. You can slide and rotate the triangles so they form a hexagon, but the hexagon is only legal if edges common to two triangles have the same number on them. You may not flip any triangle over. Two legal hexagons formed from the six triangles are illustrated below.



20+5+50+7+20+50 = 152

1+7+2+1+2+5 = 18

The score for a legal hexagon is the sum of the numbers on the outside six edges.

Your problem is to find the highest score that can be achieved with any six particular triangles.

The input file will contain one or more data sets. Each data set is a sequence of six lines with three integers from 1 to 100 separated by blanks on each line. Each line contains the numbers on the triangles in clockwise order. Data sets are separated by a line containing only an asterisk. The last data set is followed by a line containing only a dollar sign.

For each input data set, the output is a line containing only the word "none" if there are no legal hexagons or the highest score if there is a legal hexagon.

Example input:

```
1 4 20
3 1 5
50 2 3
5 2 7
7 5 20
4 7 50
*
10 1 20
20 2 30
30 3 40
40 4 50
50 5 60
60 6 10
*
10 1 20
20 2 30
30 3 40
40 4 50
50 5 60
10 6 60
$
```

Example output:

```
152
21
none
```

Last modified Tue Oct 24 20:38:19 2000

Problem B

~~Problem I~~

Crossing Prisms
Input: I.txt

Prof. Bocchan is a mathematician and a sculptor. He likes to create sculptures with mathematics.

His style to make sculptures is very unique. He uses two identical prisms. Crossing them at right angles, he makes a polyhedron that is their intersection as a new work. Since he finishes it up with painting, he needs to know the surface area of the polyhedron for estimating the amount of pigment needed.

For example, let us consider the two identical prisms in Figure 14. The definition of their cross section is given in Figure 15. The prisms are put at right angles with each other and their intersection is the polyhedron depicted in Figure 16. An approximate value of its surface area is 194.8255.

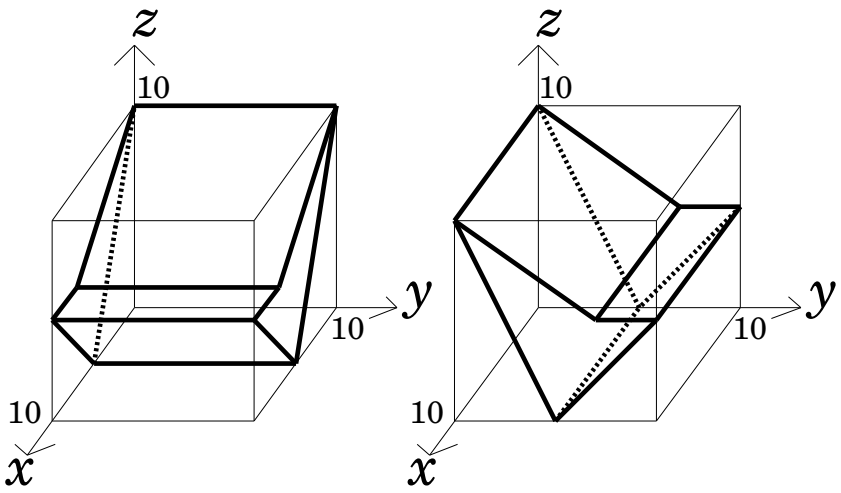


Figure 14: Two identical prisms at right angles

Given the shape of the cross section of the two identical prisms, your job is to calculate the surface area of his sculpture.

Input

The input consists of multiple datasets, followed by a single line containing only a zero. The first line of each dataset contains an integer n indicating the number of the following lines, each of which contains two integers a_i and b_i $(i = 1, \cdots, n)$.

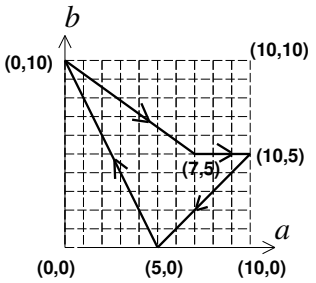


Figure 15: Outline of the cross section

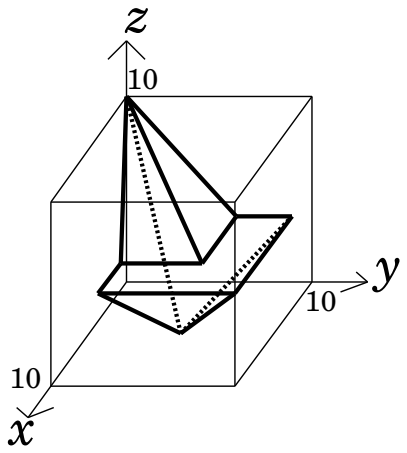


Figure 16: The intersection

A closed path formed by the given points (a_1, b_1), (a_2, b_2), $\cdots$, (a_n, b_n), $(a_{n+1}, b_{n+1})(= (a_1, b_1))$ indicates the outline of the cross section of the prisms. The closed path is simple, that is, it does not cross nor touch itself. The right-hand side of the line segment from (a_i, b_i) to (a_{i+1}, b_{i+1}) is the inside of the section.

You may assume that $3 \leq n \leq 4$, $0 \leq a_i \leq 10$ and $0 \leq b_i \leq 10$ $(i = 1, \cdots, n)$.

One of the prisms is put along the x-axis so that the outline of its cross section at $x = \xi$ is indicated by points $(x_i, y_i, z_i) = (\xi, a_i, b_i)$ $(0 \leq \xi \leq 10,\ i = 1, \cdots, n)$. The other prism is put along the y-axis so that its cross section at $y = \eta$ is indicated by points $(x_i, y_i, z_i) = (a_i, \eta, b_i)$ $(0 \leq \eta \leq 10,\ i = 1, \cdots, n)$.

Output

The output should consist of a series of lines each containing a single decimal fraction. Each number should indicate an approximate value of the surface area of the polyhedron defined by

the corresponding dataset. The value may contain an error less than or equal to 0.0001. You may print any number of digits below the decimal point.

Sample Input

```
4
5 0
0 10
7 5
10 5
4
7 5
10 5
5 0
0 10
4
0 10
10 10
10 0
0 0
3
0 0
0 10
10 0
4
0 10
10 5
0 0
9 5
4
5 0
0 10
5 5
10 10
4
0 5
5 10
10 5
5 0
4
7 1
4 1
0 1
9 5
0
```

Output for the Sample Input

```
194.8255
194.8255
600.0000
341.4214
42.9519
182.5141
282.8427
149.2470
```

Problem C
Leaky Cryptography
Input: C.txt

The ACM ICPC judges are very careful about not leaking their problems, and all communications are encrypted. However, one does sometimes make mistakes, like using too weak an encryption scheme. Here is an example of that.

The encryption chosen was very simple: encrypt each chunk of the input by flipping some bits according to a shared key. To provide reasonable security, the size of both chunk and key is 32 bits.

That is, suppose the input was a sequence of m 32-bit integers.

$$N_1 \quad N_2 \quad N_3 \ \ldots \ N_m$$

After encoding with the key K it becomes the following sequence of m 32-bit integers.

$$(N_1 \wedge K)\ (N_2 \wedge K)\ (N_3 \wedge K) \ldots (N_m \wedge K)$$

where $(a \wedge b)$ is the *bitwise exclusive or* of a and b.

Exclusive or is the logical operator which is 1 when only one of its operands is 1, and 0 otherwise. Here is its definition for 1-bit integers.

$$\begin{array}{ll} 0 \oplus 0 = 0 & 0 \oplus 1 = 1 \\ 1 \oplus 0 = 1 & 1 \oplus 1 = 0 \end{array}$$

As you can see, it is identical to addition modulo 2. For two 32-bit integers a and b, their bitwise exclusive or $a \wedge b$ is defined as follows, using their binary representations, composed of 0's and 1's.

$$a \wedge b = a_{31} \ldots a_1 a_0 \wedge b_{31} \ldots b_1 b_0 = c_{31} \ldots c_1 c_0$$

where

$$c_i = a_i \oplus b_i \ (i = 0, 1, \cdots, 31).$$

For instance, using binary notation, 11010110 $\wedge$ 01010101 = 10100011, or using hexadecimal, `d6` $\wedge$ `55` = `a3`.

Since this kind of encryption is notoriously weak to statistical attacks, the message has to be compressed in advance, so that it has no statistical regularity. We suppose that $N_1\ N_2 \ldots N_m$ is already in compressed form.

However, the trouble is that the compression algorithm itself introduces some form of regularity: after every 8 integers of compressed data, it inserts a checksum, the sum of these integers. That is, in the above input, $N_9 = \sum_{i=1}^{8} N_i = N_1 + \cdots + N_8$, where additions are modulo 2^{32}.

Luckily, you could intercept a communication between the judges. Maybe it contains a problem for the finals!

As you are very clever, you have certainly seen that you can easily find the lowest bit of the key, denoted by K_0. On the one hand, if $K_0 = 1$, then after encoding, the lowest bit of $\sum_{i=1}^{8} N_i \wedge K$ is unchanged, as K_0 is added an even number of times, but the lowest bit of $N_9 \wedge K$ is changed, so they shall differ. On the other hand, if $K_0 = 0$, then after encoding, the lowest bit of $\sum_{i=1}^{8} N_i \wedge K$ shall still be identical to the lowest bit of $N_9 \wedge K$, as they do not change. For instance, if the lowest bits after encoding are `1 1 1 1 1 1 1 1 1` then K_0 must be 1, but if they are `1 1 1 1 1 1 1 0 1` then K_0 must be 0.

So far, so good. Can you do better?

You should find the key used for encoding.

Input

The input starts with a line containing only a positive integer S, indicating the number of datasets in the input. S is no more than 1000.

It is followed by S datasets. Each dataset is composed of nine 32-bit integers corresponding to the first nine chunks of a communication. They are written in hexadecimal notation, using digits '`0`' to '`9`' and lowercase letters '`a`' to '`f`', and with no leading zeros. They are separated by a space or a newline. Each dataset is ended by a newline.

Output

For each dataset you should output the key used for encoding. Each key shall appear alone on its line, and be written in hexadecimal notation, using digits '`0`' to '`9`' and lowercase letters '`a`' to '`f`', and with no leading zeros.

Sample Input

```
8
1 1 1 1 1 1 1 1 8
3 2 3 2 3 2 3 2 6
3 4 4 7 7 b a 2 2e
e1 13 ce 28 ca 6 ab 46 a6d
b08 49e2 6128 f27 8cf2 bc50 7380 7fe1 723b
4eba eb4 a352 fd14 6ac1 eed1 dd06 bb83 392bc
ef593c08 847e522f 74c02b9c 26f3a4e1 e2720a01 6fe66007
7a4e96ad 6ee5cef6 3853cd88
60202fb8 757d6d66 9c3a9525 fbcd7983 82b9571c ddc54bab 853e52da
22047c88 e5524401
```

Output for the Sample Input

```
0
2
6
1c6
4924afc7
ffff95c5
546991d
901c4a16
```

1994 East-Central Regionals
of the ACM International Collegiate Programming Contest

Problem D

Jack Straws

In the game of Jack Straws, a number of plastic or wooden "straws" are dumped on the table and players try to remove them one-by-one without disturbing the other straws. Here, we are only concerned with if various pairs of straws are connected by a path of touching straws. You will be given a list of the endpoints for some straws (as if they were dumped on a large piece of graph paper) and then will be asked if various pairs of straws are connected. Note that touching is connecting, but also two straws can be connected indirectly via other connected straws.

Input:
A problem consists of multiple lines of input. The first line will be an integer n ($1 < n < 13$) giving the number of straws on the table. Each of the next n lines contain 4 *positive integers*, x_1, y_1, x_2 and y_2, giving the coordinates, $(x_1, y_1), (x_2, y_2)$ of the endpoints of a single straw. All coordinates will be less than 100. (Note that the straws will be of varying lengths.) The first straw entered will be known as straw #1, the second as straw #2, and so on. The remaining lines of input (except for the final line) will each contain two positive integers, a and b, both between 1 and n, inclusive. You are to determine if straw a can be connected to straw b. When $a = 0 = b$, the input is terminated. There will be no illegal input and there are no zero-length straws.

Output:
You should generate a line of output for each line containing a pair a and b, except the final line where $a = 0 = b$. The line should say simply "`CONNECTED`", if straw a is connected to straw b, or "`NOT CONNECTED`", if straw a is not connected to straw b. For our purposes, a straw is considered connected to itself.

Sample Input:

```
7
1 6 3 3
4 6 4 9
4 5 6 7
1 4 3 5
3 5 5 5
5 2 6 3
5 4 7 2
1 4
1 6
3 3
6 7
2 3
1 3
0 0
```

Sample Output:

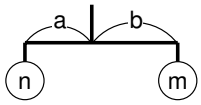
```
CONNECTED
NOT CONNECTED
CONNECTED
CONNECTED
NOT CONNECTED
CONNECTED
```

Problem E
Mobile Computing
Input: E.txt

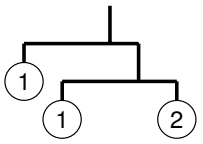
There is a mysterious planet called Yaen, whose space is 2-dimensional. There are many beautiful stones on the planet, and the Yaen people love to collect them. They bring the stones back home and make nice mobile arts of them to decorate their 2-dimensional living rooms.

In their 2-dimensional world, a mobile is defined recursively as follows:

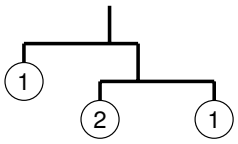
- a stone hung by a string, or
- a rod of length 1 with two sub-mobiles at both ends; the rod is hung by a string at the center of gravity of sub-mobiles. When the weights of the sub-mobiles are n and m, and their distances from the center of gravity are a and b respectively, the equation $n \times a = m \times b$ holds.



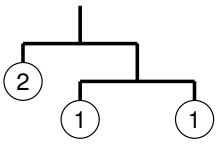
For example, if you got three stones with weights 1, 1, and 2, here are some possible mobiles and their widths:



width $= 1 + (1/3)$



width $= 1 + (2/3)$



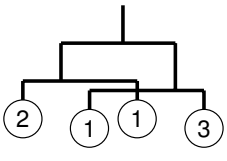
width $= 1 + (1/2)$

Given the weights of stones and the width of the room, your task is to design the widest possible mobile satisfying both of the following conditions.

- It uses all the stones.
- Its width is less than the width of the room.

You should ignore the widths of stones.

In some cases two sub-mobiles hung from both ends of a rod might overlap (see the figure on the right). Such mobiles are acceptable. The width of the example is $(1/3) + 1 + (1/4)$.



Input

The first line of the input gives the number of datasets. Then the specified number of datasets follow. A dataset has the following format.

r
s
w_1
$\vdots$
w_s

r is a decimal fraction representing the width of the room, which satisfies $0 < r < 10$. s is the number of the stones. You may assume $1 \leq s \leq 6$. w_i is the weight of the i-th stone, which is an integer. You may assume $1 \leq w_i \leq 1000$.

You can assume that no mobiles whose widths are between $r - 0.00001$ and $r + 0.00001$ can be made of given stones.

Output

For each dataset in the input, one line containing a decimal fraction should be output. The decimal fraction should give the width of the widest possible mobile as defined above. An output line should not contain extra characters such as spaces.

In case there is no mobile which satisfies the requirement, answer -1 instead.

The answer should not have an error greater than 0.00000001. You may output any number of digits after the decimal point, provided that the above accuracy condition is satisfied.

Sample Input

```
5
1.3
3
1
2
1
1.4
3
1
2
1
2.0
3
1
```

```
2
1
1.59
4
2
1
1
3
1.7143
4
1
2
3
5
```

Output for the Sample Input

```
-1
1.3333333333333335
1.6666666666666667
1.5833333333333335
1.7142857142857142
```

Problem F
Dice Puzzle
Input: F.txt

Let's try a dice puzzle. The rules of this puzzle are as follows.

1. Dice with six faces as shown in Figure 6 are used in the puzzle.

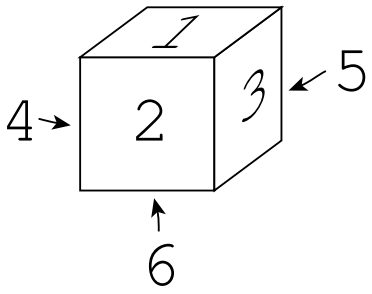


Figure 6: Faces of a die

2. With twenty seven such dice, a $3 \times 3 \times 3$ cube is built as shown in Figure 7.

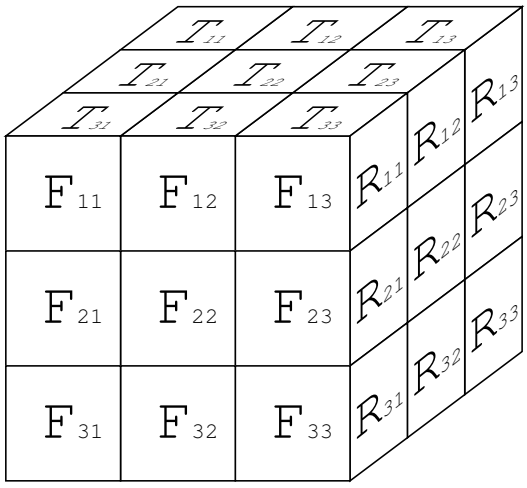


Figure 7: $3 \times 3 \times 3$ cube

3. When building up a cube made of dice, the sum of the numbers marked on the faces

of adjacent dice that are placed against each other must be seven (See Figure 8). For example, if one face of the pair is marked “2”, then the other face must be “5”.

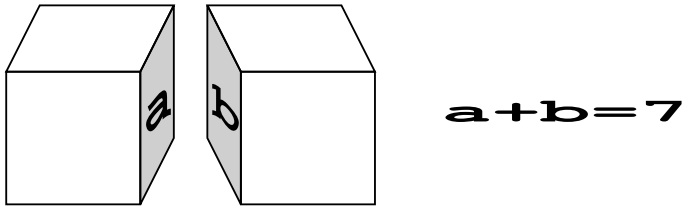


Figure 8: A pair of faces placed against each other

4. The top and the front views of the cube are partially given, i.e. the numbers on faces of some of the dice on the top and on the front are given.

T_{11}	T_{12}	T_{13}
T_{21}	T_{22}	T_{23}
T_{31}	T_{32}	T_{33}

Top

F_{11}	F_{12}	F_{13}
F_{21}	F_{22}	F_{23}
F_{31}	F_{32}	F_{33}

Front

Figure 9: Top and front views of the cube

5. The goal of the puzzle is to find all the plausible dice arrangements that are consistent with the given top and front view information.

Your job is to write a program that solves this puzzle.

Input

The input consists of multiple datasets in the following format.

N
$Dataset_1$
$Dataset_2$
...
$Dataset_N$

N is the number of the datasets.

The format of each dataset is as follows.

T_{11} T_{12} T_{13}
T_{21} T_{22} T_{23}
T_{31} T_{32} T_{33}
F_{11} F_{12} F_{13}
F_{21} F_{22} F_{23}
F_{31} F_{32} F_{33}

T_{ij} and F_{ij} ($1 \leq i \leq 3$, $1 \leq j \leq 3$) are the faces of dice appearing on the top and front views, as shown in Figure 7, or a zero. A zero means that the face at the corresponding position is unknown.

Output

For each plausible arrangement of dice, compute the sum of the numbers marked on the nine faces appearing on the right side of the cube, that is, with the notation given in Figure 7, $\sum_{i=1}^{3} \sum_{j=1}^{3} R_{ij}$.

For each dataset, you should output the right view sums for all the plausible arrangements, in ascending order and without duplicates. Numbers should be separated by a single space.

When there are no plausible arrangements for a dataset, output a zero.

For example, suppose that the top and the front views are given as follows.

1	0	0
0	2	0
0	0	0

Top view

5	1	2
5	1	2
0	0	0

Front view

Figure 10: Example

There are four plausible right views as shown in Figure 11. The right view sums are 33, 36, 32, and 33, respectively. After rearranging them into ascending order and eliminating duplicates, the answer should be "32 33 36".

4	4	4
4	4	4
3	3	3

4	4	4
4	4	4
4	4	4

3	4	4
3	4	4
4	3	3

3	4	4
3	4	4
3	4	4

Figure 11: Plausible right views

The output should be one line for each dataset. The output may have spaces at ends of lines.

Sample Input

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

Output for the Sample Input

```
27
24
32 33 36
0
```

Problem G
Color the Map
Input: G.txt

You were lucky enough to get a map just before entering the legendary magical mystery world. The map shows the whole area of your planned exploration, including several countries with complicated borders. The map is clearly drawn, but in sepia ink only; it is hard to recognize at a glance which region belongs to which country, and this might bring you into severe danger. You have decided to color the map before entering the area. "A good deal depends on preparation," you talked to yourself.

Each country has one or more territories, each of which has a polygonal shape. Territories belonging to one country may or may not "touch" each other, i.e. there may be disconnected territories. All the territories belonging to the same country must be assigned the same color. You can assign the same color to more than one country, but, to avoid confusion, two countries "adjacent" to each other should be assigned different colors. Two countries are considered to be "adjacent" if any of their territories share a border of non-zero length.

Write a program that finds the least number of colors required to color the map.

Input

The input consists of multiple map data. Each map data starts with a line containing the total number of territories n, followed by the data for those territories. n is a positive integer not more than 100. The data for a territory with m vertices has the following format:

$$
\begin{array}{l}
\mathit{String} \\
x_1 \ \ y_1 \\
x_2 \ \ y_2 \\
\cdots \\
x_m \ \ y_m \\
-1
\end{array}
$$

"*String*" (a sequence of alphanumerical characters) gives the name of the country it belongs to. A country name has at least one character and never has more than twenty. When a country has multiple territories, its name appears in each of them.

Remaining lines represent the vertices of the territory. A vertex data line has a pair of nonnegative integers which represent the x- and y-coordinates of a vertex. x- and y-coordinates are separated by a single space, and y-coordinate is immediately followed by a newline. Edges of the territory are obtained by connecting vertices given in two adjacent vertex data lines, and by

connecting vertices given in the last and the first vertex data lines. None of x- and y-coordinates exceeds 1000. Finally, -1 in a line marks the end of vertex data lines. The number of vertices m does not exceed 100.

You may assume that the contours of polygons are simple, i.e. they do not cross nor touch themselves. No two polygons share a region of non-zero area. The number of countries in a map does not exceed 10.

The last map data is followed by a line containing only a zero, marking the end of the input data.

Output

For each map data, output one line containing the least possible number of colors required to color the map satisfying the specified conditions.

Sample Input

```
6
Blizid
0 0
60 0
60 60
0 60
0 50
50 50
50 10
0 10
-1
Blizid
0 10
10 10
10 50
0 50
-1
Windom
10 10
50 10
40 20
20 20
20 40
10 50
-1
Accent
50 10
50 50
```

```
35 50
35 25
-1
Pilot
35 25
35 50
10 50
-1
Blizid
20 20
40 20
20 40
-1
4
A12345678901234567 89
0 0
0 100
100 100
100 0
-1
B12345678901234567 89
100 100
100 200
200 200
200 100
-1
C12345678901234567 89
0 100
100 100
100 200
0 200
-1
D12345678901234567 8
100 0
100 100
200 100
200 0
-1
0
```

Output for the Sample Input

```
4
2
```

Problem H
Inherit the Spheres
Input: H.txt

In the year 2xxx, an expedition team landing on a planet found strange objects made by an ancient species living on that planet. They are transparent boxes containing opaque solid spheres (Figure 12). There are also many lithographs which seem to contain positions and radiuses of spheres.

Figure 12: A strange object

Initially their objective was unknown, but Professor Zambendorf found the cross section formed by a horizontal plane plays an important role. For example, the cross section of an object changes as in Figure 13 by sliding the plane from bottom to top.

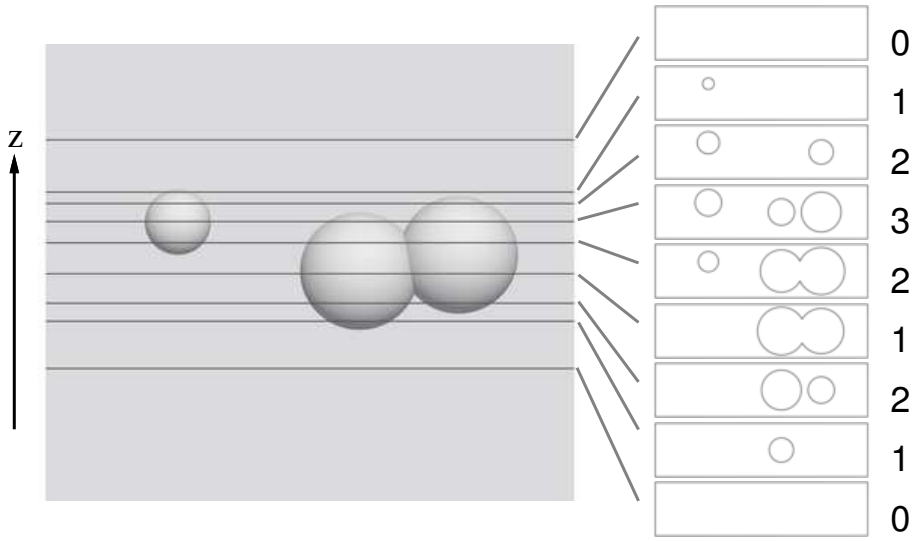


Figure 13: Cross sections at different positions

He eventually found that some information is expressed by the transition of the number of connected figures in the cross section, where each connected figure is a union of discs intersecting or touching each other, and each disc is a cross section of the corresponding solid sphere. For instance, in Figure 13, whose geometry is described in the first sample dataset later, the number of connected figures changes as 0, 1, 2, 1, 2, 3, 2, 1, and 0, at $z = 0.0000$, 162.0000, 167.0000, 173.0004, 185.0000, 191.9996, 198.0000, 203.0000, and 205.0000, respectively. By assigning 1 for increment and 0 for decrement, the transitions of this sequence can be expressed by an 8-bit binary number 11011000.

For helping further analysis, write a program to determine the transitions when sliding the horizontal plane from bottom ($z = 0$) to top ($z = 36000$).

Input

The input consists of a series of datasets. Each dataset begins with a line containing a positive integer, which indicates the number of spheres N in the dataset. It is followed by N lines describing the centers and radiuses of the spheres. Each of the N lines has four positive integers X_i, Y_i, Z_i, and R_i ($i = 1, \cdots, N$) describing the center and the radius of the i-th sphere, respectively.

You may assume $1 \leq N \leq 100$, $1 \leq R_i \leq 2000$, $0 < X_i - R_i < X_i + R_i < 4000$, $0 < Y_i - R_i < Y_i + R_i < 16000$, and $0 < Z_i - R_i < Z_i + R_i < 36000$. Each solid sphere is defined as the set of all points (x, y, z) satisfying $(x - X_i)^2 + (y - Y_i)^2 + (z - Z_i)^2 \leq R_i^2$.

A sphere may contain other spheres. No two spheres are mutually tangent. Every $Z_i \pm R_i$ and minimum/maximum z coordinates of a circle formed by the intersection of any two spheres differ from each other by at least 0.01.

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

Output

For each dataset, your program should output two lines. The first line should contain an integer M indicating the number of transitions. The second line should contain an M-bit binary number that expresses the transitions of the number of connected figures as specified above.

Sample Input

```
3
95 20 180 18
125 20 185 18
40 27 195 10
1
5 5 5 4
2
5 5 5 4
5 5 5 3
2
5 5 5 4
5 7 5 3
16
2338 3465 29034 710
1571 14389 25019 842
1706 8015 11324 1155
1899 4359 33815 888
2160 10364 20511 1264
2048 8835 23706 1906
2598 13041 23679 618
1613 11112 8003 1125
1777 4754 25986 929
2707 9945 11458 617
1153 10358 4305 755
2462 8450 21838 934
1822 11539 10025 1639
1473 11939 12924 638
1388 8519 18653 834
2239 7384 32729 862
0
```

Output for the Sample Input

```
8
11011000
2
10
2
10
2
10
28
10111001001101011010001011 00
```